On the Notion of Plan Quality for PDDL+

Francesco Percassi[1], **Enrico Scala**[2], **Mauro Vallati**[1]

[1] School of Computing and Engineering, University of Huddersfield, Huddersfield, United Kingdom
[2] Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Brescia, Italy
f.percassi@hud.ac.uk, enrico.scala@unibs.it, m.vallati@hud.ac.uk

Abstract

PDDL+ is a planning formalism designed to model mixed continuous-discrete problems. Despite its expressiveness, the absence of a well-established framework for evaluating plan quality makes it challenging to use PDDL+ in applications where plan shape and quality are crucial.

This paper addresses this issue by introducing a comprehensive set of plan cost functions tailored for discrete-time PDDL+, along with a cost-preserving translation for generating cost-aware PDDL2.1 planning tasks. The plan cost functions provide a theoretical ground for assessing plan quality, whereas the translation shows their practicability by leveraging the connection between PDDL+ and PDDL2.1.

Introduction

Automated planning addresses the problem of synthesising sequences of actions to achieve a goal, starting from an initial state and following a formal model. The planning community has designed several formalisms to represent realistic problems (Cesta and Oddi 1996; McDermott et al. 1998; Fox and Long 2003; Sanner 2010). Among these, PDDL+ stands out as one of the most expressive (Fox and Long 2006). Inspired by hybrid automata (Henzinger 1996; Henzinger et al. 1998), PDDL+ allows the modelling of hybrid discrete-continuous problems that incorporate numeric and temporal features (Bonassi, Gerevini, and Scala 2024).

A key feature that sets PDDL+ apart from other planning formalisms is its capability to blend agent-oriented actions with environmental dynamics through *processes* and *events*. These elements differ in their nature: actions represent *potential transitions* controlled by the agent, while *processes* and *events* involve *must transitions* that occur when certain logical conditions are met. More in detail, processes describe how the system's numeric variables continuously evolve over time based on ordinary differential equations, while events represent instantaneous changes.

The modelling capabilities of PDDL+ have promoted the application of automated planning in a wide range of challenging real-world applications (Alaboud and Coles 2019; Kiam et al. 2020; Cardellini et al. 2021; Aineto et al. 2023; Alon et al. 2024; Kouaiti et al. 2024) and even physics-based games (Piotrowski et al. 2023). However, despite its expressive power, the challenge of evaluating plan quality in a domain-independent manner remains underexplored. A significant limitation is the lack of a well-established framework for evaluating plan quality, which hampers the ability to generate quality-aware plans. This is reflected in the lack of explicit support for plan quality in state-of-the-art domain-independent PDDL+ planning engines. For instance, SMTPLAN+ (Cashmore, Magazzeni, and Zehtabi 2020) encodes PDDL+ problems into a satisfiability modulo theory formulation, focusing solely on goal achievement without considering plan quality. ENHSP (Scala et al. 2016), although originally designed for numeric planning, extends its heuristics to PDDL+ in an ad-hoc manner that does not fully capture the language's specific features. UPMURPHI (Penna, Magazzeni, and Mercorio 2012) relies on explicit model checking and exhaustive search, prioritising completeness and goal achievement over optimisation; however, its breadth-first strategy often produces plans with near-optimal makespan. DINO (Piotrowski et al. 2016) builds upon UPMURPHI by introducing heuristic guidance to improve scalability, but it also lacks explicit mechanisms for optimising plan quality or makespan. Overall, no planner explicitly manages plan quality. However, heuristic-based planners tend to be biased against makespan by treating wait actions, which correspond to an agent's decision to let time pass, as if they were domain actions. A notable exception is the work by Say (2023), although it is not framed in PDDL+.

To address this gap, we formalise a comprehensive set of cost functions designed for PDDL+ plans. Here we focus on discrete-time PDDL+, a key approach commonly used by planning engines and practical applications to manage the complexity of planning tasks (Penna, Magazzeni, and Mercorio 2012; Fox, Long, and Magazzeni 2012; Piotrowski et al. 2016; Piotrowski and Perez 2024).

In particular, the plan cost functions we introduce fall into two categories. The first comprises extensions of numeric and temporal planning cost functions (Fox and Long 2003), which naturally align with the language syntax. These functions, however, have not yet been formally defined in this context. The second comprises functions inspired by control theory, reflecting the influence of hybrid automata on the language design (Bogomolov et al. 2014, 2015).

To study the practicability of these plan cost functions, we

illustrate how to extend an existing translation of discrete-time PDDL+ planning tasks into equivalent PDDL2.1 ones (Percassi, Scala, and Vallati 2023), making them *cost-aware*. This contribution shows that the proposed cost functions can be exploited to guide plan construction, enabling optimisation during planning rather than post-hoc evaluation.

Background

In this section, we provide the necessary background on discrete-time PDDL+, numeric planning, and the approach for solving PDDL+ via translation to numeric planning.

Discrete-Time PDDL+

A *discrete-time* PDDL+ *planning task* Π is a tuple $\langle F, X, I, G, A, E, P\rangle$ along with a *time discretisation step* $\delta \in \mathbb{Q}^+$, where each element is defined as follows. F and X are finite sets of Boolean and numeric variables. A *state* s is a total mapping from F to $\{\bot, \top\}$ and from X to $\mathbb{Q}$. Given a set of variables V, we denote by $S(V)$ the set of all total assignments over V. I is one of these states referred to as the initial state. Boolean variables can be involved in Boolean conditions $\langle v = b\rangle$, where $v \in F$ and $b \in \{\top, \bot\}$. Numeric variables can be involved in numeric conditions $\langle \xi \bowtie 0\rangle$, where ξ is an arithmetic expression over X and $\bowtie \in \{>, \geq, =, \leq, <\}$. We denote by $\mathcal{L}$ the language of propositional formulae built over Boolean and numeric conditions as atomic predicates. $G \in \mathcal{L}$ is the goal state description. A, E and P are finite sets of actions, events and processes, respectively. Each element $z \in A \cup E \cup P$ is described as a pair $\langle \mathit{pre}(z), \mathit{eff}(z)\rangle$, where $\mathit{pre}(z)$ is the precondition of z and $\mathit{eff}(z)$ is the effect associated with z. The preconditions $\mathit{pre}(z)$ are elements from $\mathcal{L}$. For actions and events, the effect $\mathit{eff}(z)$ consists of a set of Boolean and numeric effects. A *Boolean effect* has the form of $\langle v := b\rangle$ where $v \in F$ and $b \in \{\bot, \top\}$. A *numeric effect* has the form $\langle v := \xi\rangle$ where $v \in X$ and ξ is an arithmetic expression. (For convenience, we use two equivalent forms: $\langle \mathsf{incr}, v, \xi'\rangle$ for $\langle v := v + \xi'\rangle$, and $\langle \mathsf{asgn}, v, \xi\rangle$ for $\langle v := \xi\rangle$.) The effects of processes are described as sets of *discrete-time numeric effects*, expressed as pairs $\langle v, \xi\rangle$, where $v \in X$ and ξ is an arithmetic expression.[1] We will use a, ρ, and ε to refer to a generic action, process, and event, respectively.

A PDDL+ *plan* π_t is a pair $\langle \pi, t_e\rangle$, where $\pi = \langle\langle a_1, t_1\rangle, \ldots, \langle a_n, t_n\rangle\rangle$ is a sequence of timestamped actions from A, and $t_e \in \mathbb{Q}_0^+$ is the plan's makespan. A plan π_t is well-defined w.r.t. δ if for every $i \in \{1, \ldots, n-1\}$, $0 \leq t_i \leq t_{i+1} \leq t_e$ holds and t_i, t_e are multiple of δ.

Let s be a state, $v \in F \cup X$ and ξ an arithmetic expression, we denote with $s[v]$ the value assumed by v in s, and with $s[\xi]$ the evaluation of ξ in s. A state s satisfies a Boolean condition, written as $s \models \langle v = b\rangle$, iff $s[v] = b$. A state s satisfies a numeric condition written as $s \models \langle \xi \bowtie 0\rangle$ iff $s[\xi] \bowtie 0$. Building on these atomic definitions, the satisfaction of a formula $p \in \mathcal{L}$ by a state s, written $s \models p$, follows the standard semantics of propositional logic. Applying an action a in a state s yields a new state $s' = \gamma(s, \{a\})$ where:

$$s'[v] = \begin{cases} b & \text{if } \langle v := b\rangle \in \mathit{eff}(a), v \in F \\ s[\xi] & \text{if } \langle v := \xi\rangle \in \mathit{eff}(a), v \in X \\ s[v] & \text{otherwise.} \end{cases}$$

An action a is applicable in state s if $s \models \mathit{pre}(a)$ and it has no conflicting effects. Since actions and events share the same syntactic structure, we also use the notation $\gamma(s, \{\varepsilon_1, \ldots, \varepsilon_k\})$ to denote the state resulting from sequentially applying the events in the set, in any order.

Plan validity relies on the notions of plan projection and time points, inspired by the work of Shin and Davis (2005). A *time point* $T = \langle t = \delta \cdot n, n'\rangle$ has clock t and counter n', where $n, n' \in \mathbb{N}$. The counter is used to order actions or events occurring at time t. Time points are lexicographically ordered. A *plan projection* $\mathbb{H}$ collects all transitions induced by a plan π_t, indexed over an ordered set of significant time points (STPs) $T_{\mathbb{H}} = \{T_0, \ldots, T_m\}$. $\mathbb{H}$ maps each $T \in T_{\mathbb{H}}$ to a *situation* $\mathbb{H}(T) = \langle \mathbb{H}_A(T), \mathbb{H}_s(T)\rangle$, where $\mathbb{H}_A(T)$ is the action (if any) executed at T, and $\mathbb{H}_s(T)$ is the associated state. From state $\mathbb{H}_s(T)$, we define the set of *triggered events* as $\mathcal{E}(T) = \{\varepsilon \in E \mid \mathbb{H}_s(T) \models \mathit{pre}(\varepsilon)\}$. We assume that Π is *event-deterministic*, meaning that applying triggered events in any order leads to the same outcome, and has *finite complexity*, meaning that all event cascades terminate in a finite number of steps. In line with the work by Fox, Howey, and Long (2005), this can be ensured by assuming that mutually exclusive events are never triggered simultaneously, that events self-deactivate, and that each event is triggered at most once per timestamp. We define the set of active processes at T as $\mathcal{C}(T) = \{\rho \in P \mid \mathbb{H}_s(T) \models \mathit{pre}(\rho)\}$, referred to as the *context* of T. The context of a state determines which processes are active at a given time point, guiding the evolution of numeric variables.

Algorithm 1: Plan Projection construction.

```
1:  function PlanProjection(π_t = ⟨π, t_e⟩, Π, δ)
2:      T ← ⟨0, 0⟩
3:      ℍ(T) ← ⟨∅, I⟩
4:      ExecuteEvents()
5:      for each ⟨a_i, t_i⟩ in π do
6:          while T.clock() ≠ t_i do
7:              TimeFlow()
8:          ℍ_A(T) ← a_i
9:          ℍ(T.IncrStep()) ← ⟨∅, γ(ℍ_s(T), {a_i})⟩
10:         ExecuteEvents()
11:     while T.clock() ≠ t_e do
12:         TimeFlow()
13:     return ℍ
14: procedure ExecuteEvents()
15:     while ℰ(T) ≠ ∅ do
16:         ℍ(T.IncrStep()) ← ⟨∅, γ(ℍ_s(T), ℰ(T))⟩
17: procedure TimeFlow()
18:     ℍ(T.IncrTime(δ)) ← ⟨∅, 𝒰(ℍ_s(T), 𝒞(T))⟩
19:     ExecuteEvents()
```

[1] Specifically, ξ represents the additive contribution to the finite difference of v over a time step δ. With a slight abuse of notation, this means that $v(t + \delta) = v(t) + \xi(t) \cdot \delta$.

We define the plan projection of plan π_t using Algorithm 1. $\mathbb{H}$ is initialised with the initial state I, i.e., $\langle \mathbb{H}_A(T) = \varnothing, \mathbb{H}_s(T) = I \rangle$, where $\varnothing$ indicates *no action applied*. If the clock t of T, $T.\text{CLOCK}()$, differs from the action's timestamp t_i, we advance it by δ until they match, leading to a *temporal transition*; if they match, we apply action a_i, resulting in an *instantaneous transition*. The TIMEFLOW() procedure handles temporal transitions. For $T = \langle t, n \rangle$, advancing the clock by δ and resetting the counter yields a new STP, i.e., $T.\text{INCRTIME}(\delta) = \langle t + \delta, 0 \rangle$. The new state is determined by the *update transition function* $s' = \mathcal{U}(s, \mathcal{C})$ (Line 18). In particular, for each $v \in F$, $s'[v] = s[v]$, as Boolean variables are not affected by processes, while for each $v \in X$:

$$s'[v] = s[v] + \sum_{\substack{\langle \hat{v}, \xi \rangle \in \mathit{eff}(\rho) \\ \text{s.t. } \hat{v}=v,\ \rho \in \mathcal{C}}} s[\xi \cdot \delta]. \tag{1}$$

Applying an action at T sets $\mathbb{H}_A(T) = a_i$ (Line 9), increments the counter to $T.\text{INCRSTEP}() = \langle t, n+1 \rangle$, and produces a new state. After each temporal transition or action, the EXECUTEEVENTS() procedure checks for triggered events. If none are triggered, it terminates without creating new STPs. Otherwise, it generates a new STP via an *instantaneous transition*, based on the state $\gamma(s, \mathcal{E}(T))$ (Line 16). (Note that, under the assumption that Π is event-deterministic, the outcome of $\gamma(s, \mathcal{E}(T))$ is uniquely determined.) This process iterates until a fixed point is reached, at which no further events are triggered.

After constructing the plan projection, we assess the validity of π_t. The plan is valid if, for every $T \in T_{\mathbb{H}}$ such that $\mathbb{H}_A(T) = a$, the action a is applicable in $\mathbb{H}_s(T)$, and the final state $\mathbb{H}_s(T_m)$ satisfies the goal G.

Numeric Planning with Costs

PDDL2.1 (level 2) (Fox and Long 2003) is a fragment of PDDL+ in which neither processes nor events can be specified. Its original formulation allows for flexible definitions of plan metrics, enabling the specification of any arithmetic expression involving numeric variables for optimisation, either through maximisation or minimisation. To avoid scenarios where there may be no optimal plan, such as when an action can unboundedly maximise a function, and in line with the approach of many numeric planners (Hoffmann 2003; Gerevini, Saetti, and Serina 2008; Scala et al. 2016; Li et al. 2018; Leofante et al. 2020; Kuroiwa et al. 2022), we focus on metrics that can be translated into a state-dependent cost minimisation problem.

Definition 1 (PDDL2.1 Planning Task with Costs)**.** *A* PDDL2.1 planning task with costs *is a tuple* $\langle \Pi, c \rangle$ *where* $\Pi = \langle F, X, I, G, A \rangle$ *and* $c : A \times S(X) \mapsto \mathbb{Q}_0^+$ *is a state-dependent action cost-function.*

A valid plan for $\langle \Pi, c \rangle$ is a sequence of actions $\pi = \langle a_1, \ldots, a_n \rangle$, where each $a_i \in A$, such that there exists a sequence of states $\langle s_0, \ldots, s_n \rangle$ satisfying $s_0 = I$; for each $i \in \{1, \ldots, n\}$, $s_{i-1} \models \mathit{pre}(a_i)$ and $s_i = \gamma(s_{i-1}, \{a_i\})$; $s_n \models G$. The cost of π is defined as $\mathit{cost}(\pi) = \sum_{i=1}^{n} c(a_i, s_{i-1})$. π is an optimal plan if and only if, for each valid plan π' for $\langle \Pi, c \rangle$, $\mathit{cost}(\pi) \leq \mathit{cost}(\pi')$.

Solving PDDL+ via Numeric Planning

Percassi, Scala, and Vallati (2023) proposed a method for translating discrete-time PDDL+ into PDDL2.1 by emulating processes and events through actions. Particularly, they introduced two translations, EXP and POLY, whose names refer to the size of the resulting tasks.

This work leverages EXP, summarised below. We chose EXP as it is simpler to present our results, and we believe the more feasible POLY translation cannot support some of the plan cost functions in the general case, as we will discuss later. In the following, we assume $\delta \in \mathbb{Q}^+$ given as an input parameter.

Definition 2 (EXP Translation)**.** *Let* Π *be a* PDDL+ *planning task,* EXP *generates the new* PDDL2.1 *planning task* $\Pi_{\text{EXP}} = \langle F \cup \{\mathit{fe}\}, X, I \cup \{\mathit{fe}\}, G \wedge \neg \mathit{fe}, A' \rangle$*, where each element from* A' *is detailed as follows.*

$$
\begin{aligned}
A' &= A_\Pi \cup A_{\text{SIM}} \cup A_{\text{SIMEV}} \cup \{a_{\mathit{satu}}\} \\
A_\Pi &= \{a_\Pi = \langle \mathit{pre}(a) \wedge \neg \mathit{fe}, \mathit{eff}(a) \cup \{\mathit{fe}\} \rangle \mid a \in A\} \\
A_{\text{SIM}} &= \{a_{\text{SIM}}^{\mathcal{C}} \mid \mathcal{C} \in \mathscr{P}^+(P)\} \\
\mathit{pre}(a_{\text{SIM}}^{\mathcal{C}}) &= \bigwedge_{\rho \in P \setminus \mathcal{C}} \neg \mathit{pre}(\rho) \wedge \bigwedge_{\rho \in P \cap \mathcal{C}} \mathit{pre}(\rho) \wedge \neg \mathit{fe} \\
\mathit{eff}(a_{\text{SIM}}^{\mathcal{C}}) &= \{\langle \mathsf{incr}, v, \sum_{\substack{\langle \hat{v}, \xi \rangle \in \mathit{eff}(\rho) \\ \text{s.t. } \hat{v}=v,\ \rho \in \mathcal{C}}} \xi \cdot \delta \rangle \mid v \in X\} \\
A_{\text{SIMEV}} &= \{a_{\text{SIMEV}}^{\mathcal{E}} \mid \mathcal{E} \in \mathscr{P}^+(E)\} \\
\mathit{pre}(a_{\text{SIMEV}}^{\mathcal{E}}) &= \bigwedge_{\varepsilon \in E \setminus \mathcal{E}} \neg \mathit{pre}(\varepsilon) \wedge \bigwedge_{\varepsilon \in E \cap \mathcal{E}} \mathit{pre}(\varepsilon) \wedge \mathit{fe} \\
\mathit{eff}(a_{\text{SIMEV}}^{\mathcal{E}}) &= \bigcup_{\varepsilon \in E \cap \mathcal{E}} \mathit{eff}(\varepsilon) \\
a_{\mathit{satu}} &= \langle \bigwedge_{\varepsilon \in E} \neg \mathit{pre}(\varepsilon) \wedge \mathit{fe}, \{\neg \mathit{fe}\} \rangle
\end{aligned}
$$

Π_{EXP} is extended with the set of actions A_{SIM} for handling the simulation of temporal transitions, and $A_{\text{SIMEV}} \cup \{a_{\mathit{satu}}\}$ for handling events triggering.

All the actions in A_{SIM} correspond to the agent's decision to move time forward (*waiting*) and allow this to happen according to the current context. Specifically, an action $a_{\text{SIM}}^{\mathcal{C}}$ can be executed in a state if its context is equal to $\mathcal{C}$. We enumerate all the possible contexts by examining the power set of P excluding the empty one, i.e., $\mathscr{P}^+(P) = \mathscr{P}(P) \setminus \{\emptyset\}$.

Events may be triggered following the execution of an action or the passage of time. To handle this, the variable *fe* (force events) is set to true whenever an action from $A_\Pi \cup A_{\text{SIM}}$ is executed. Once set, it blocks the execution of actions from $A_\Pi \cup A_{\text{SIM}}$ and mandates the execution of the sequence of actions $\langle \text{SIMEV} \times k, a_{\mathit{satu}} \rangle$ for handling the (possible) cascades of events. Here, $\text{SIMEV} \times k$ denotes a sequence of k actions from A_{SIMEV} where, under our assumptions concerning the events, $k \in \{0, \ldots, |E|\}$ represents the length of the event cascade. Each application of $a_{\text{SIMEV}}^{\mathcal{E}}$ deals with a set of events $\mathcal{E}$ triggered simultaneously. Similarly to the processes, $\mathcal{E} \in \mathscr{P}^+(E)$.

Finally, a_{satu} is applied when there are no more triggered events to handle, restoring the applicability of actions from $A_\Pi \cup A_{\text{SIM}}$. We remark that $k = 0$ when no events are triggered, making a_{satu} immediately applicable, while $k = |E|$ when all events trigger in cascade, one at a time.

PDDL+ with a Plan Cost Function

In this section, we formalise PDDL+ with an objective function to minimise, i.e., a *plan cost function*. We introduce four plan cost functions: makespan (MS), an arithmetic expression (ψ), roughness (RN), and swiftness for $\tau \in \mathbb{Q}^+$ (SW(τ)). MS and ψ can be syntactically defined in PDDL+, though a precise formulation for ψ is yet to be established. Here, they are reframed for the discrete-time context.

Definition 3 (PDDL+ Planning Tasks with Cost Function)**.** *A* PDDL+ planning task with cost function *is a tuple* $\Pi^{\mathfrak{C}} = \langle \Pi, \mathfrak{C} \rangle$ *where* Π *is a* PDDL+ *task and* $\mathfrak{C} \in \{\psi, \text{MS}, \text{RN}, \text{SW}(\tau)\}$ *is an objective function to minimise.*

Given a plan π_t for $\Pi^{\mathfrak{C}}$, its cost is denoted as $cost^{\mathfrak{C}}(\pi_t)$. π_t is an optimal plan if and only if, for each valid plan π'_t for $\Pi^{\mathfrak{C}}$, $cost^{\mathfrak{C}}(\pi_t) \leq cost^{\mathfrak{C}}(\pi'_t)$. Before addressing the formal definitions, we introduce the running example that will be used to showcase differences in plans induced by $\mathfrak{C}$.

Running Example

We consider the PDDL+ domain LINEAR-GENERATOR. The goal is to run the generator for 1000 units of time, i.e., $d_{\mathit{run}} = 1000$. However, the initial fuel is insufficient for this task, but it can be replenished from the tanks $\mathsf{T} \in \mathsf{Tanks}$. The running and refuelling activities are modelled as processes, i.e., ρ_{run} and $\rho_{\mathit{ref}(\mathsf{T})}$. These processes start and end through pairs of actions and events. Refuel has a flexible duration, expressed through $d_{\mathit{ref}(\mathsf{T})} \leq 10$. The agent can initiate a process by executing the actions

$$a_{\mathit{run}} = \langle \neg \mathit{run}, \{\mathit{run}, \langle \mathsf{asgn}, \theta_{\mathit{run}}, 0 \rangle\} \rangle$$
$$a_{\mathit{ref}(\mathsf{T})} = \langle \neg \mathit{ref}(\mathsf{T}) \wedge \neg \mathit{done}_{\mathit{ref}(\mathsf{T})}, \{\mathit{ref}(\mathsf{T}), \langle \mathsf{asgn}, \theta_{\mathit{ref}(\mathsf{T})}, 0 \rangle\} \rangle.$$

Both actions set the clock to track how much time the related processes remain active, i.e., θ_{run} and $\theta_{\mathit{ref}(\mathsf{T})}$. Further, they make true the Boolean variables *run* and $\mathit{ref}(\mathsf{T})$ activating the processes

$$\rho_{\mathit{run}} = \langle \mathit{run} \wedge \langle \mathit{fuel} > 0 \rangle, \{\langle \mathit{fuel}, -1 \rangle, \langle \theta_{\mathit{run}}, 1 \rangle\} \rangle$$
$$\rho_{\mathit{ref}(\mathsf{T})} = \langle \mathit{ref}(\mathsf{T}), \{\langle \mathit{fuel}, 1 \rangle, \langle \theta_{\mathit{ref}(\mathsf{T})}, 1 \rangle, \langle \mathit{fuelDrawn}, 1 \rangle\} \rangle.$$

When ρ_{run} ($\rho_{\mathit{ref}(\mathsf{T})}$) is active, it decreases (increases) fuel with a unitary linear rate according to $\langle \mathit{fuel}, -1 \rangle \in \mathit{eff}(\rho_{\mathit{run}})$ ($\langle \mathit{fuel}, 1 \rangle \in \mathit{eff}(\rho_{\mathit{ref}(\mathsf{T})})$). Both processes increment their respective clocks while processes $\rho_{\mathit{ref}(\mathsf{T})}$ track how much fuel has been drawn according to $\langle \mathit{fuelDrawn}, 1 \rangle \in \mathit{eff}(\rho_{\mathit{ref}(\mathsf{T})})$. If the process ρ_{run} lasts for d_{run}, a special event marking the achievement of the goal is triggered:

$$\varepsilon_{\downarrow \mathit{run}} = \langle \mathit{run} \wedge \langle \theta_{\mathit{run}} = 1000 \rangle, \{\neg \mathit{run}, \mathit{achieved}\} \rangle.$$

Since the refuelling activities have a flexible duration, the agent has two possibilities for stopping them. In the first case, it can wait until the maximum duration is reached, causing the triggering of the event $\varepsilon_{\downarrow \mathit{ref}(\mathsf{T})}$. Otherwise, the agent can intentionally deactivate the process early using the action $a_{\downarrow \mathit{ref}(\mathsf{T})}$. $\varepsilon_{\downarrow \mathit{ref}(\mathsf{T})}$ and $a_{\downarrow \mathit{ref}(\mathsf{T})}$ are defined as

$$\varepsilon_{\downarrow \mathit{ref}(\mathsf{T})} = \langle \mathit{ref}(\mathsf{T}) \wedge \langle \theta_{\mathit{ref}(\mathsf{T})} \geq 10 \rangle, \{\neg \mathit{ref}(\mathsf{T}), \mathit{done}_{\mathit{ref}(\mathsf{T})}\} \rangle$$
$$a_{\downarrow \mathit{ref}(\mathsf{T})} = \langle \mathit{ref}(\mathsf{T}) \wedge \langle \theta_{\mathit{ref}(\mathsf{T})} < 10 \rangle, \{\neg \mathit{ref}(\mathsf{T}), \mathit{done}_{\mathit{ref}(\mathsf{T})}\} \rangle.$$

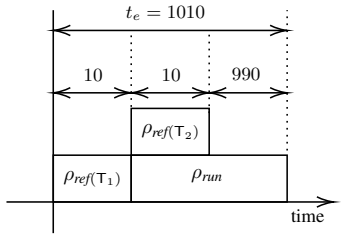


Figure 1: Plan π_t for Π_{Gen}. (Left) Graphical representation of the plan (process durations are not in scale for readability). (Right) Timestamped actions and events of π_t.

Further, to prevent the fuel quantity from exceeding the generator's capacity, an event for modelling the overflow is defined as $\varepsilon_{\mathit{of}} = \langle \langle \mathit{fuel} > 1000 \rangle \wedge \neg \mathit{of}, \{\mathit{of}\} \rangle$. If $\varepsilon_{\mathit{of}}$ is triggered, the goal becomes unachievable. Finally, we can express the goal as $G = \neg \mathit{run} \wedge \bigwedge_{\mathsf{T} \in \mathsf{Tanks}} \neg \mathit{ref}(\mathsf{T}) \wedge \mathit{achieved} \wedge \neg \mathit{of}$. This means that no activity must be ongoing, the goal must be achieved, and no overflow must have occurred.

We consider the planning task Π_{Gen} with two tanks $\mathsf{Tanks} = \{\mathsf{T}_1, \mathsf{T}_2\}$ and where the initial fuel level in the generator is 984. A valid plan π_t for Π_{Gen} discretised in $\delta = 1$ is shown in Figure 1. The left side displays a graphical representation of the plan over time, highlighting the activities performed by the processes. The right side lists the timed actions of the plan in the format "t_i : action". Events are included.

Makespan (MS)

As PDDL+ is an approach to reasoning about time, the most intuitive plan cost function to consider is the plan's duration. This is the primary objective in PDDL+ and temporal planning, and is of interest in hybrid planning as well (Chen, Williams, and Fan 2021). Minimising the makespan would be beneficial in human-robot interaction scenarios, where shorter plans facilitate the human's recognition of operations (Capitanelli et al. 2018) and in traffic control applications, where there is the need to efficiently move a specific volume of vehicles in the least amount of time (Percassi et al. 2023).

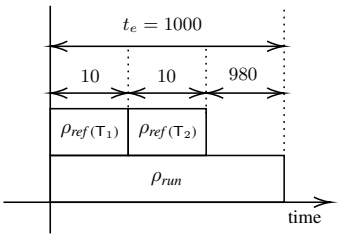
Given a valid plan $\pi_t = \langle \pi, t_e \rangle$ for Π^{MS}, its cost can be simply defined as $cost^{\text{MS}}(\pi_t) = t_e$.

Makespan-Optimal Plan We couple the running example with the plan cost function MS, defining $\Pi^{\text{MS}}_{\mathsf{Gen}}$, and evaluate the plan accordingly. Unsurprisingly, plan π_t (Figure 1) is suboptimal; its duration is 1010 time units, i.e., $cost^{\text{MS}}(\pi_t) = 1010$, because the first refuel activity involving T_1 is completed before the generator run has started.

An MS-optimal plan π^{MS}_t is provided in Figure 2. This plan contextually starts the first refuelling and the generator, leading to a total duration of 1000 time units.

Arithmetic Expression (ψ)

In this section, we introduce a cost function based on arbitrary arithmetic expressions ψ, defined over the numeric variables of the planning task. Optimising ψ enables the



$0.0 : a_{run}$
$0.0 : a_{ref(\mathsf{T}_1)}$
$10.0 : \varepsilon_{\downarrow ref(\mathsf{T}_1)}$
$10.0 : a_{ref(\mathsf{T}_2)}$
$20.0 : \varepsilon_{\downarrow ref(\mathsf{T}_2)}$
$1000.0 : \varepsilon_{\downarrow run}$

Figure 2: Optimal plan π_t^{MS} for $\Pi^{\text{MS}}_{\mathsf{Gen}}$ having $cost^{\text{MS}}(\pi_t^{\text{MS}}) = 1000$. (Left) Graphical representation of the plan. (Right) Timestamped actions and events of π_t.

specification of diverse cost criteria, such as energy consumption or resource usage, which may not be directly related to makespan. For instance, in the PDDL+ formulation of the Urban Traffic Control problem (Percassi et al. 2023), one can define a numeric variable, *wastedGreen*, which increases whenever a green phase is active at an intersection, but cannot be exploited, either because no vehicles are present in the feeder link, or because the receiver link is congested. Minimising *wastedGreen* encourages traffic light phases that reduce ineffective green time, potentially leading to improved throughput in ways that differ from simply minimising makespan.

In the following, we formally define how the cost of a plan can be evaluated for a given expression ψ, and how it can be computed incrementally by analysing the state transitions in the plan projection. Similarly to the numeric case, ψ can be syntactically any arithmetic expression. However, we focus on cases where the transition costs induced by ψ are non-negative. In practice, as illustrated in the traffic control and generator examples, this is ensured by defining ψ over variables whose values increase monotonically over time, such as cumulative quantities.

Let π_t be a valid plan for Π^ψ. To define its cost given ψ, we compute the plan projection $\mathbb{H}$ and evaluate the value of ψ in the final state, i.e., $cost^\psi(\pi_t) = \mathbb{H}_s(T_m)[\psi]$, where T_m is the last STP of $\mathbb{H}$. Nevertheless, the plan cost can be computed incrementally, as in PDDL2.1, by considering all transitions (events, actions, and processes) that affect ψ. To formally define this, we need some additional notation.

Given an action a, we denote by $\mathit{eff}_\psi(a) \subseteq \mathit{eff}(a)$ those effects of a which affect ψ. Events and processes happen in parallel, so we reason in terms of context $\mathcal{C} \subseteq P$ and triggered events $\mathcal{E} \subseteq E$. For processes, given a context $\mathcal{C}$, we denote by $\mathit{eff}_\psi(\mathcal{C})$ the discrete-time effects of $\mathcal{C}$ affecting ψ (collected according to Equation 1), and similarly, for a set of triggered events $\mathit{eff}_\psi(\mathcal{E})$. Finally, we denote by $\psi_{|\mathit{eff}_\psi(z)}$ the expression obtained by applying all effects in $\mathit{eff}_\psi(z)$ to ψ, that is, by replacing every variable v occurring in ψ with the corresponding right-hand side ξ from each assignment $\langle v := \xi \rangle$ in $\mathit{eff}_\psi(z)$. Here, z may refer to an action, a context, or a set of events. By exploiting these definitions, we can define a state-dependent "cost" function $c^\psi : A \cup \mathscr{P}^+(E) \cup \mathscr{P}^+(P) \times S(X) \mapsto \mathbb{Q}$ as

$$c^\psi(z, s) = (\psi_{|\mathit{eff}_\psi(z)} - \psi)[s], \tag{2}$$

where the cost of z is calculated as the difference between the value of ψ after the transition caused by z and its value before the transition. This definition does not depend on the particular form of ψ or the effects involved, and without making further assumptions, there can be situations where no optimal plans exist. Therefore, similarly to PDDL2.1, we require c^ψ to only produce non-negative costs, i.e., $c^\psi : A \cup \mathscr{P}^+(E) \cup \mathscr{P}^+(P) \times S(X) \mapsto \mathbb{Q}_0^+$.

To calculate the plan cost, we enumerate all the STPs associated with a plan projection $\mathbb{H}$, i.e., $T_\mathbb{H}$, and for each of them, sum the contribution provided by Equation 2. To do this it is essential to distinguish, from the ordered set of STPs $T_\mathbb{H}$, which refers to a temporal transition.

Definition 4 (Temporal Transitions Set)**.** *Let π_t be a* PDDL+ *plan for Π and $\mathbb{H}$ its plan projection defined over $T_\mathbb{H} = \{T_0, \ldots, T_m\}$. The* temporal transitions set of $\mathbb{H}$ *is defined as the ordered set* $\Theta(\mathbb{H}) = \{T_i \in T_\mathbb{H} \mid i \in \{0, \ldots, m-1\}, T_i.\textsc{IncrTime}(\delta) = T_{i+1}\}$.

Intuitively, we collect all STPs whose successors are ahead in time by a duration δ.

Definition 5 (PDDL+ Plan Cost for Π^ψ)**.** *Let π_t be a* PDDL+ *plan for Π^ψ, $\mathbb{H}$ its plan projection defined over $T_\mathbb{H} = \{T_0, \ldots, T_m\}$ and $\Theta(\mathbb{H})$ the temporal transitions set of $\mathbb{H}$ as for Definition 4. Then:*

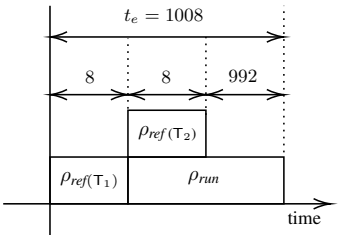
$$cost^\psi(\pi_t) = \sum_{i=0}^{m} c^\psi(\mathcal{Z}(\mathbb{H}, T_i), \mathbb{H}_s(T_i)), \text{ where}$$

$$\mathcal{Z}(\mathbb{H}, T) = \begin{cases} \mathcal{C}(T) & \text{if } T \in \Theta(\mathbb{H}) \\ \mathcal{E}(T) & \text{if } \mathbb{H}_A(T) = \varnothing \\ \mathbb{H}_A(T) & \text{otherwise.} \end{cases}$$

Intuitively, the formula calculates the cost of a plan by summing the individual costs associated with each STP T in $T_\mathbb{H}$. The function $\mathcal{Z}(\mathbb{H}, T)$ acts as a selector that determines which element from $A \cup \mathscr{P}^+(E) \cup \mathscr{P}^+(P)$ refers to T. If T corresponds to a temporal transition, $\mathcal{Z}$ selects the context of T; if no action is associated with T, it chooses the events triggered by T; otherwise, it selects the action $\mathbb{H}_A(T) = a$.

ψ-Optimal Plan We consider the planning task from the running example and define the cost function to minimise the amount of fuel drawn from the tanks, that is, Π^ψ_{Gen} with $\psi = \mathit{fuelDrawn}$. The only elements in the planning task affecting ψ are the processes $\rho_{ref(\mathsf{T}_1)}$ and $\rho_{ref(\mathsf{T}_2)}$. In particular, by applying Equation 2, we compute the cost associated with the temporal transitions induced by the contexts $\{\rho_{ref(\mathsf{T}_1)}\}$, $\{\rho_{ref(\mathsf{T}_2)}\}$, and their combination. Since these processes affect *fuelDrawn* with a constant linear rate, the cost of each corresponding temporal transition is constant and independent of the state in which it occurs. Thereby, we have $c^\psi(\{\rho_{ref(\mathsf{T})}\}, s) = \delta = 1$ for both tanks, independently of the state s. If refuelling involves both tanks simultaneously, then $c^\psi(\{\rho_{ref(\mathsf{T}_1)}, \rho_{ref(\mathsf{T}_2)}\}, s) = 2$. The plans π_t and π_t^{MS} in Figures 1-2 are both suboptimal for ψ as the refuelling activities last for the maximum duration. Thus, $cost^\psi(\pi_t) = cost^\psi(\pi_t^{\text{MS}}) = 20$.

Figure 3 shows a ψ-optimal plan, π_t^ψ, with minimised refuelling durations. Since $I[\mathit{fuel}] = 984$, each refuelling must



$0.0 : a_{ref(\mathsf{T}_1)}$
$8.0 : a_{\downarrow ref(\mathsf{T}_1)}$
$8.0 : a_{ref(\mathsf{T}_2)}$
$8.0 : a_{run}$
$16.0 : a_{\downarrow ref(\mathsf{T}_2)}$
$1008.0 : \varepsilon_{\downarrow run}$

Figure 3: Optimal plan π_t^ψ for Π^ψ_{Gen} having $cost^\psi(\pi_t^\psi) = 16$. (Left) Graphical representation of the plan. (Right) Timestamped actions and events from π_t^ψ.

last at least 8 time units to reach the goal. Notably, this plan terminates refuelling using the actions $a_{\downarrow ref(T)}$ rather than the events $\varepsilon_{\downarrow ref(T)}$, which are triggered upon reaching the maximum duration. Consequently, $cost^\psi(\pi_t^\psi) = 16$.

Roughness (RN) and Swiftness (SW(τ))

Plan cost functions can be defined by drawing inspiration from the concept of *average dwell time* (ADT), which refers to the average time a hybrid automaton remains in a specific *mode of operation* before switching to another mode (Morse 1996; Hespanha and Morse 1999). ADT is widely used in the design of stable controllers, ensuring that a system stays in a mode for at least τ time units, thereby limiting frequent switching that can lead to instability (Mitra and Liberzon 2004). In PDDL+, we propose an analogous concept where a *context*, defined as a set of active processes, represents the system's current mode. A change in context, induced by the activation or deactivation of processes, reflects a shift in system dynamics. To adapt ADT to PDDL+, we define two plan cost functions: *roughness* and *swiftness*.

Roughness (RN) counts the number of context switches in a plan, encouraging plans with fewer dynamic transitions. The more changes, the higher the RN. A plan with high RN is one where the system keeps shifting its behaviour, jumping from one dynamic to another, while a low-RN plan stays steady longer. To illustrate the relevance of this notion, we sketch some potential applications in domains where frequent changes are undesirable and stability is important. For example, in traffic signal optimisation, low-RN plans produce stable signal configurations, avoiding frequent changes that are often unacceptable to authorities and challenging to implement due to infrastructure constraints (Kouaiti et al. 2024). Similarly, in railway planning, frequent switches in engine modes (e.g., accelerating, braking) can reduce the mechanical lifespan of costly equipment. By minimising RN, plans reduce mechanical stress.

While RN is inspired by the idea of ADT, it does not incorporate an explicit temporal dimension; to more closely capture the timing aspects central to ADT, we introduce a complementary plan cost function called *swiftness*. Given a temporal threshold $\tau > 0$, *Swiftness* (SW(τ)) penalises contexts that persist for less than τ time units, encouraging longer durations between switches. This reflects the spirit of ADT in hybrid systems, which ensures stability by requiring that, on average, switches do not occur too frequently.

Roughness (RN) The first definition we need is the set of STPs in $T_{\mathbb{H}}$ associated with a context switch.

Definition 6 (Context Switch Set)**.** *Let $\mathbb{H}$ be a plan projection and $\Theta(\mathbb{H}) = \{T'_0, \ldots, T'_\theta\}$ the temporal transition set of $\mathbb{H}$. The* context switch set of $\mathbb{H}$ *is defined as* $\Lambda(\mathbb{H}) = \{T'_i \in \Theta(\mathbb{H}) \mid i \in \{1, \ldots, \theta\}, \mathcal{C}(T'_i) \neq \mathcal{C}(T'_{i-1})\}$.

Intuitively, $\Lambda(\mathbb{H})$ is defined as the set of all STPs T'_i in $\Theta(\mathbb{H})$ such that the context at T'_{i-1} differs from that at T'_i.

Definition 7 (PDDL+ Plan Cost for Π^{RN})**.** *Let $\pi_t = \langle \pi, t_e \rangle$ be a* PDDL+ *plan for Π^{RN}, $\mathbb{H}$ its plan projection and $\Lambda(\mathbb{H})$ the context switch set of $\mathbb{H}$. Then:*

$$cost^{\text{RN}}(\pi_t) = \begin{cases} 0 & \text{if } t_e = 0 \\ 1 + |\Lambda(\mathbb{H})| & \text{otherwise.} \end{cases}$$

We assume that the first temporal transition is a context switch, so when the makespan is greater than 0, the minimum roughness is 1. For plans with no duration (including the empty plan), roughness is 0.

Swiftness (SW(τ)) To define the cost of a plan for SW(τ), we must extend the set $\Lambda(\mathbb{H})$ by adding the initial and ending STPs of $\mathbb{H}$, i.e., T_0 and T_m. We denote such an ordered set as $\overline{\Lambda}(\mathbb{H}) = \{T_0\} \cup \Lambda(\mathbb{H}) \cup \{T_m\} = \{\langle \hat{T}_0, \ldots, \hat{T}_\lambda \rangle\}$.

Definition 8 (PDDL+ Plan Cost for $\Pi^{\text{SW}(\tau)}$)**.** *Let π_t be a* PDDL+ *plan for $\Pi^{\text{SW}(\tau)}$, $\mathbb{H}$ its plan projection and $\overline{\Lambda}(\mathbb{H}) = \{\langle \hat{t}_0, \cdot \rangle, \ldots, \langle \hat{t}_\lambda, \cdot \rangle\}$ the extended set of $\Lambda(\mathbb{H})$. Then:*

$$cost^{\text{SW}(\tau)}(\pi_t) = \sum_{i \in \{1,\ldots,\lambda\}} [\hat{t}_i - \hat{t}_{i-1} < \tau],$$

where $[P]$ is the Iverson bracket that returns 1 when condition P is true, 0 otherwise.

For example, consider two consecutive STPs in $\overline{\Lambda}(\mathbb{H})$, i.e., $\hat{T}_{i-1} = \langle \hat{t}_{i-1}, \cdot \rangle$ and $\hat{T}_i = \langle \hat{t}_i, \cdot \rangle$. The duration of the active context at $\hat{T}_{i-1}$ before it changes at $\hat{T}_i$ is calculated as $\hat{t}_i - \hat{t}_{i-1}$. By counting the instances where this duration is below τ, we obtain the plan cost for SW(τ).

Roughness-Optimal Plan We consider the planning task from the running example, where the objective is to minimise RN, denoted as $\Pi^{\text{RN}}_{\mathsf{Gen}}$. Figures 4a-4b show a RN-optimal plan, π_t^{RN}, where all refuelling activities are parallelised to reduce the number of context switches. Below, we explain how the cost of π_t^{RN} can be derived using the definitions introduced earlier.

We begin by constructing the plan projection $\mathbb{H}$ from π_t^{RN} and deriving the corresponding $T_{\mathbb{H}}$ using Algorithm 1. Figure 4c illustrates $\mathbb{H}$ and $T_{\mathbb{H}}$. The x-axis shows all STPs from $T_{\mathbb{H}}$. A single arrow represents an instantaneous transition caused by either an action or an event, denoted as ITA and ITE, respectively. For example, $T_0 = \langle 0, 0 \rangle$ corresponds to an ITA from the application of $\langle a_{run}, 0 \rangle$, producing $T_1 = \langle 0, 1 \rangle$. Similarly, $T_{13} = \langle 10, 0 \rangle$ corresponds to an ITE triggered by $\langle \varepsilon_{\downarrow ref(\mathsf{T}_1)}, 10 \rangle$, resulting in $T_{14} = \langle 10, 1 \rangle$. A double arrow indicates a sequence of temporal transitions (TTs), during which a context persists.

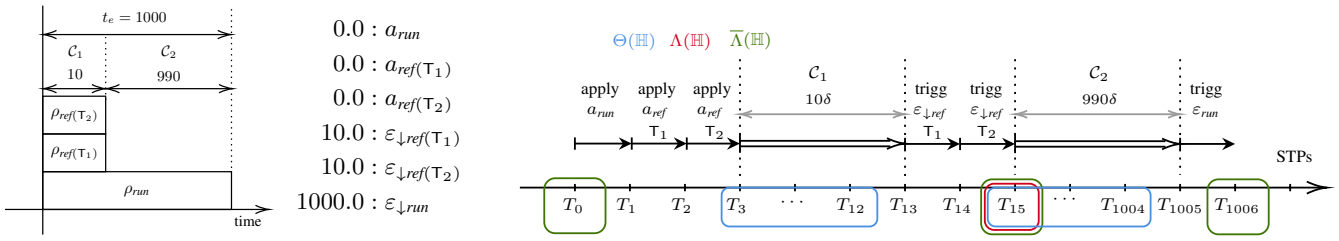


Figure 4: Optimal plan π_t^{RN} for $\Pi_{\text{Gen}}^{\text{RN}}$ having $cost^{\text{RN}}(\pi^{\text{RN}}) = 2$. (a) Graphical representation of the plan. (b) Timestamped actions and events of π_t^{RN}. (c) Graphical depictions of $\mathbb{H}$ and $T_{\mathbb{H}}$ induced by π_t^{RN}. Arrows indicate ITAs and ITEs. Double arrows denote sequences of TTs with the same context. STPs from $\Theta(\mathbb{H})$ are highlighted in blue, $\Lambda(\mathbb{H})$ in red and $\overline{\Lambda}(\mathbb{H})$ in green.

For instance, over the time interval $(0, 10)$, linked to STPs $\{T_3, \ldots, T_{13}\}$, the context driving the numeric variables is $\mathcal{C}_1 = \{\rho_{run}, \rho_{ref(\mathsf{T}_1)}, \rho_{ref(\mathsf{T}_2)}\}$.

Overall, π_t^{RN} generates three ITAs, three ITEs, and 1000 TTs, resulting in $T_{\mathbb{H}} = \{T_0, \ldots, 1006\}$. By applying Definition 4, we restrict $T_{\mathbb{H}}$ to obtain the set of STPs associated with TTs, i.e., $\Theta(\mathbb{H}) = \{T_3, \ldots, T_{12}\} \cup \{T_{15}, \ldots, T_{1004}\}$. From this set, the only STP involving a context switch is T_{15}, i.e., $\Lambda(\mathbb{H}) = \{T_{15}\}$. Specifically, $\mathcal{C}_1 = \mathcal{C}(T_{12}) \neq \mathcal{C}(T_{15}) = \mathcal{C}_2 = \{\rho_{run}\}$ due to the end of the refuelling activities. The new context, $\mathcal{C}_2$, persists throughout the time interval $(10, 1000)$, corresponding to STPs $\{T_{15}, \ldots, T_{1004}\}$. Now, we can compute the cost of π_t^{RN} for RN as $cost^{\text{RN}}(\pi_t^{\text{RN}}) = 1 + |\Lambda(\mathbb{H})| = 2$. Interestingly, in this domain, minimising RN naturally promotes the parallelisation of activities.

Finally, we calculate the plan cost for $\text{SW}(\tau)$ when $\tau = 10$. First, we extend $\Lambda(\mathbb{H})$ to include the initial and final STPs of $\mathbb{H}$, resulting in $\overline{\Lambda}(\mathbb{H}) = \{T_0, T_{15}, T_{1006}\}$. Using Definition 8, we observe that π_t^{RN} is also $\text{SW}(\tau)$-optimal because both contexts $\mathcal{C}_1$ and $\mathcal{C}_2$ persist for at least τ. Specifically, $\mathcal{C}_1$ persists for $t_{15} - t_0 = 10 \geq \tau$, and $\mathcal{C}_2$ for $t_{1006} - t_{15} = 990 \geq \tau$. Thus, $cost^{\text{SW}(\tau)}(\pi_t^{\text{RN}}) = 0$.

Cost-Preserving Translations to PDDL2.1

This section describes how to translate a PDDL+ task with a cost function into an *equivalent* PDDL2.1 task with costs, in a cost-preserving fashion. This enables the use of cost-optimal numeric planners for PDDL+ tasks. The translation EXP (Definition 2) serves as the basis.

Given a plan cost function $\mathfrak{C}$ and a translation Z from discrete-time PDDL+ to PDDL2.1, we denote by $Z^{\mathfrak{C}}$ the variant of Z incorporating $\mathfrak{C}$. Given $\Pi^{\mathfrak{C}}$, we denote by $\Gamma_Z^{\mathfrak{C}}$ the numeric planning task obtained by applying $Z^{\mathfrak{C}}$ to $\Pi^{\mathfrak{C}}$. We refer to a plan for $\Gamma_Z^{\mathfrak{C}}$ as $\pi_Z^{\mathfrak{C}}$.

Definition 9 (Cost-Preserving Translation)**.** *A translation Z is* cost-preserving for $\mathfrak{C}$ *if, for every task $\Pi^{\mathfrak{C}}$, there is a bijection between its valid plans and those of the corresponding task $\Gamma_Z^{\mathfrak{C}}$, such that both have the same cost.*

To ease the following proofs about the cost-preservingness of the $\mathfrak{C}$-aware variants of EXP, we leverage Theorem 1 provided by Percassi, Scala, and Vallati (2023). Theorem 1 proves the soundness and completeness of EXP through a proof by construction showing how to get a valid plan π_t for Π into one, namely π_{EXP}, valid for Π_{EXP} (and vice versa). A by-product of the proof is that we have the state trajectories induced by plans $\langle \pi_t, \pi_{\text{EXP}} \rangle$ equivalent over $F \cup X$. This is the outcome of the following lemma.

Lemma 1. *Let Π be a* PDDL+ *planning task and Π_{EXP} the* PDDL2.1 *planning task obtained by applying* EXP*. Then, there exists a bijection between the valid plans of Π and Π_{EXP} such that τ, the state trajectory generated by collecting each state from the plan projection of π_t, and τ', the state trajectory generated by iteratively applying the plan π_{EXP}, are equivalent over $F \cup X$.*

Proof. The proof of the Lemma relies on the operational construction of a mapping between the plans of Π and Π_{EXP}, presented below.

(From $\pi_t = \langle \pi, t_e \rangle$ to π_{EXP}) For all actions $\langle a, t \rangle$ from π, we add the corresponding compiled action $a_\Pi \in A_\Pi$ in π_{EXP} following the original order given by π. Furthermore, we append to every a_Π the sequence $\langle \text{SIMEV} \times k, a_{satu} \rangle$ simulating the events triggering. For each temporal transition induced by π_t, we add the sequence $\langle a_{\text{SIM}}^{\mathcal{C}} \rangle + \langle \text{SIMEV} \times k, a_{satu} \rangle$ simulating the effects of both processes and events.

(From π_{EXP} to π_t) For each action a_Π in π_{EXP}, which belongs to A_Π, we add the timestamped action $\langle a, t \rangle$ in π_t, where $a \in A$ is the original action from which a_Π has been derived, and t is equal to the number of actions from A_{SIM} preceding a_Π multiplied by δ. As previously done, we achieve this by preserving the order of actions as in π_{EXP}. Similarly, the number of actions from A_{SIM} in π_{EXP} multiplied by δ gives us the makespan t_e. We consider $\langle \pi_t, \pi_{\text{EXP}} \rangle$ for Π and Π_{EXP}, constructed as above, *equivalent*. The demonstration of the equivalence of the state trajectories induced by π_t and π_{EXP} is a direct consequence of the mapping, and the full proof can be found in Theorem 1 from the work by Percassi, Scala, and Vallati (2023). □

Makespan in PDDL2.1

The EXP translation can be easily adapted to handle the makespan minimisation. Given a PDDL+ planning task Π^{MS}, the translation EXP^{MS} generates the PDDL2.1 planning task with costs $\Gamma_{\text{EXP}}^{\text{MS}} = \langle \Pi_{\text{EXP}}, c_{\text{EXP}}^{\text{MS}} \rangle$ in which Π_{EXP} is generated according to Definition 2 and $c_{\text{EXP}}^{\text{MS}}(a) = \delta$ if $a \in A_{\text{SIM}}$, 0 otherwise. Since the actions from A_{SIM} are the

ones that advance time, they have a cost equal to δ, while all the others have zero cost.

Theorem 1. EXP^{MS} *is cost-preserving w.r.t.* MS.

Proof Sketch. It is easy to see that the total cost of the plan $\pi^{\text{MS}}_{\text{EXP}}$ is equal to t_e, as each temporal transition from π_t corresponds to multiple occurrences of an action from A_{SIM}, each with a cost of δ. The total number of occurrences is $\frac{t_e}{\delta}$. □

ψ in PDDL2.1

Similarly to the makespan case, handling ψ through translation can be achieved by extending Π_{EXP} with an action cost function, which in this case must be *state-dependent.*

Definition 10 (Translation EXP^{ψ})**.** *Let* Π^{ψ} *be a* PDDL+ *planning task,* EXP^{ψ} *generates a* PDDL2.1 *planning task* $\Gamma^{\psi}_{\text{EXP}} = \langle \Pi_{\text{EXP}}, c^{\psi}_{\text{EXP}} \rangle$ *where* Π_{EXP} *is defined as for Definition 2, and* c^{ψ}_{EXP} *is a state-dependent cost function defined for an action* $a^{\star}$ *and a state* s *as*

$$c^{\psi}_{\text{EXP}}(a^{\star}, s) = \begin{cases} c^{\psi}(a, s) & \text{if } a^{\star} = a_{\Pi} \in A_{\Pi} \\ c^{\psi}(\mathcal{E}, s) & \text{if } a^{\star} = a^{\mathcal{E}}_{\text{SIMEV}} \in A_{\text{SIMEV}} \\ c^{\psi}(\mathcal{C}, s) & \text{if } a^{\star} = a^{\mathcal{C}}_{\text{SIM}} \\ 0 & \text{if } a^{\star} = a_{\mathit{satu}}. \end{cases}$$

Referring to the above definition, the function c^{ψ}_{EXP} extends the cost model defined by c^{ψ} (see Equation 2) to the compiled actions in the PDDL2.1 encoding generated by the EXP translation. It distinguishes among four categories of compiled actions.

First, when $a^{\star}$ is a compiled action $a_{\Pi} \in A_{\Pi}$, derived from an action $a \in A$ in the original PDDL+ task, the cost is directly inherited from the original model as $c^{\psi}(a, s)$. Second, when $a^{\star}$ is a compiled action $a^{\mathcal{E}}_{\text{SIMEV}}$ handling a set of triggered events $\mathcal{E}$, its cots is defined as $c^{\psi}(\mathcal{E}, s)$. Third, if $a^{\star}$ is a wait action $a^{\mathcal{C}}_{\text{SIM}}$ associated with a context $\mathcal{C}$, then its cost is given by $c^{\psi}(\mathcal{C}, s)$. Finally, since a_{satu} is a spurious action that does not affect the ψ, its cost is 0.

Theorem 2. EXP^{ψ} *is cost-preserving w.r.t.* ψ.

Proof Sketch. Using Lemma 1, we can show that for any pair of equivalent plans $\langle \pi_t, \pi^{\psi}_{\text{EXP}} \rangle$ for Π^{ψ} and $\Gamma^{\psi}_{\text{EXP}}$, the cost of π_t is equal to that of π^{ψ}_{EXP}. In particular, for every action $\langle a, t \rangle$ of π_t, there exists a corresponding action in π^{ψ}_{EXP} applied in an equivalent state with respect to $F \cup X$ due to Lemma 1. This guarantees that the action costs are equal. Furthermore, for every temporal transition induced by π_t with context $\mathcal{C}(T)$, there is a matching action $a^{\mathcal{C}}_{\text{SIM}}$ in π^{ψ}_{EXP}. Since the compiled task ensures that only the action $a^{\mathcal{C}(T)}_{\text{SIM}}$ is applicable in that context, the temporal transition and $a^{\mathcal{C}(T)}_{\text{SIM}}$ produce the same cost. The same argument applies to instantaneous transitions associated with triggered events. This leads us to conclude that $cost^{\psi}(\pi_t) = cost(\pi^{\psi}_{\text{EXP}})$. □

A natural question arises as to whether the same cost extension can be applied to POLY, the polynomial encoding proposed by Percassi, Scala, and Vallati (2023), in which processes are simulated sequentially. Currently, we do not have an answer to this question, as defining a cost-aware version of this translation would require prior knowledge of the cost associated with each temporal transition in a given context. However, we believe that such an extension may be possible for specific restricted fragments of ψ, and we plan to explore this direction as part of our future work.

Roughness and Swiftness in PDDL2.1

To address RN and $\text{SW}(\tau)$, it is not sufficient to adjust the cost function for the actions added by EXP; a more structural change in the translation is required. Our approach for both RN and $\text{SW}(\tau)$ involves tracking the current and previous context during each temporal transition to detect context switches. For RN, we increase the cost only if the two contiguous contexts differ, which we achieve by adding two mutually applicable actions. We will primarily focus on the translation for RN and use it as a foundation for $\text{SW}(\tau)$.

Definition 11 (Translation EXP^{RN})**.** *Let* Π^{RN} *be a* PDDL+ *planning task,* EXP^{RN} *generates a* PDDL2.1 *planning task* $\Gamma^{\text{RN}}_{\text{EXP}} = \langle \langle F', X', I \cup \{\mathit{fe}\}, G', A' \rangle, c^{\text{RN}}_{\text{EXP}} \rangle$ *where each element is detailed as follows.*

F'	$= F \cup \{h, \mathit{syn}, \mathit{fe}\}$
X'	$= X \cup X_P \cup X^o_P$
X_P	$= \{x_\rho \mid \rho \in P\} \quad X^o_P = \{x^o_\rho \mid \rho^o \in P\}$
G'	$= G \wedge \neg h \wedge \neg \mathit{syn} \wedge \neg \mathit{fe}$
A'	$= A'' \cup A_{\text{RN-SIM}} \cup \{a^{=}, a^{\neq}, a_{\mathit{copy}}\}$
A''	$= \{\langle \mathit{pre}(a) \wedge \neg h, \mathit{eff}(a) \rangle \mid a \in A_{\Pi} \cup A_{\text{SIMEV}} \cup \{a_{\mathit{satu}}\}\}$
$A_{\text{RN-SIM}}$	$= \{a^{\mathcal{C}}_{\text{RN-SIM}} = \langle \mathit{pre}(a^{\mathcal{C}}_{\text{SIM}}) \wedge \neg h, \mathit{eff}(a^{\mathcal{C}}_{\text{SIM}}) \cup \{\mathit{syn}, h\} \cup$ $\{\langle \mathsf{asgn}, x_\rho, [\rho \in \mathcal{C}] \rangle \mid \rho \in P\} \rangle \mid a^{\mathcal{C}}_{\text{SIM}} \in A_{\text{SIM}}\}$
$a^{=}$	$= \langle \bigwedge_{\rho \in P} \langle x_\rho = x^o_\rho \rangle \wedge \mathit{syn}, \{\neg \mathit{syn}\} \rangle$
$a^{\neq}$	$= \langle \bigvee_{\rho \in P} \neg \langle x_\rho = x^o_\rho \rangle \wedge \mathit{syn}, \{\neg \mathit{syn}\} \rangle$
a_{copy}	$= \langle \neg \mathit{syn} \wedge h, \{\langle \mathsf{asgn}, x^o_\rho, x_\rho \rangle \mid \rho \in P\} \cup \{\neg h\} \rangle$

The sets A_{Π}*,* A_{SIM}*,* A_{SIMEV} *and* $\{a_{\mathit{satu}}\}$ *are defined as Definition 2, while the action cost function is defined as* $c^{\text{RN}}_{\text{EXP}}(a) = 1$ *when* $a = a^{\neq}$*, 0 otherwise.*

EXP^{RN} extends X by adding $X_P \cup X^o_P$, which includes numeric variables x_ρ and x^o_ρ for each process $\rho \in P$. These variables, with a domain of $\{0, 1\}$, indicate whether a process is active (1) or inactive (0). For each simulated temporal transition, X_P and X^o_P store the *current* and *previous* context, respectively, in binary form.

The actions previously in A_{SIM} are now part of $A_{\text{RN-SIM}}$ with two functions: (i) advancing time, and (ii) storing the current context into X_P. Here, $[\rho \in \mathcal{C}]$ is an Iverson bracket that returns 1 if $\rho \in \mathcal{C}$, and 0 otherwise.

To move time forward, the agent must execute the triplet $\langle a^{\mathcal{C}}_{\text{RN-SIM}}, a^{\star}, a_{\mathit{copy}} \rangle$, where $a^{\star} \in \{a^{=}, a^{\neq}\}$ based on whether a context switch has just occurred. This specific order is enforced by Boolean variables *syn* (synchronise) and *h* (halt). The action $a^{\mathcal{C}}_{\text{RN-SIM}}$ updates the numeric variables, according to $\mathcal{C}$, and X_P to reflect active processes in the current

temporal transition. The second action, $a^\star$, signals whether the context has changed. If the context matches the previous one, $a^\star = a^=$ (condition entailed by $\bigwedge_{\rho \in P} \langle x_\rho = x^o_\rho \rangle$). If the context has changed, $a^\star = a^\neq$ ($\bigvee_{\rho \in P} \neg \langle x_\rho = x^o_\rho \rangle$). The final action, a_{copy}, copies the current context stored in X_P into X^o_P to track future temporal transitions.

All actions have a cost of 0, except for $a^\neq$, which is executed whenever a context switch occurs between consecutive temporal transitions and incurs a unitary cost.

Theorem 3. EXP$^{\text{RN}}$ *is cost-preserving w.r.t.* RN.

Proof Sketch. First, (i) we show that Lemma 1 for EXP generalises to EXP$^{\text{RN}}$ by defining a pair of equivalent plans for this translation. Then, (ii) we show the cost equivalence.

For (i) let π_t be a plan for Π^{RN}. The equivalent plan $\pi^{\text{RN}}_{\text{EXP}}$ for $\Gamma^{\text{RN}}_{\text{EXP}}$ is derived using the same mapping as for EXP, except that the time-advancing action $a^{\mathcal{C}}_{\text{SIM}}$ is replaced by a triplet $\langle a^{\mathcal{C}}_{\text{RN-SIM}}, a^\star, a_{copy} \rangle$. Mapping from $\pi^{\text{RN}}_{\text{EXP}}$ to π_t uses the same approach as for EXP, omitting the actions $\{a^=, a^\neq, a_{copy}\}$. Since the extensions introduced by EXP$^{\text{RN}}$ do not affect $F \cup X$, any valid plan for Π^{RN} still corresponds to a valid one for $\Gamma^{\text{RN}}_{\text{EXP}}$ and vice versa. Moreover, if we collapse the triplet $\langle a^{\mathcal{C}}_{\text{RN-SIM}}, a^\star, a_{copy} \rangle$ into a single transition, we observe that Lemma 1 generalises for EXP$^{\text{RN}}$ as $a^\star$ and a_{copy} do not interfere with variables $F \cup X$.

(ii) The cost of π_t is determined by the number of context switches (plus one) while $\pi^{\text{RN}}_{\text{EXP}}$ depends on the occurrences of $a^\neq$. As shown for (i), the numeric variables remain synchronised throughout the traces of π_t and $\pi^{\text{RN}}_{\text{EXP}}$ as the machinery actions do not affect them. This ensures that, for each contiguous pair of temporal transitions induced by π_t, with the previous context $\mathcal{C}^o$ and the current context $\mathcal{C}$, the variables X^o_P and X_P track the binary representations of $\mathcal{C}^o$ and $\mathcal{C}$ after applying the corresponding action $a^{\mathcal{C}}_{\text{RN-SIM}}$, and vice versa. This implies that every context switch between $\mathcal{C}^o$ and $\mathcal{C}$ will always be associated with $a^\star = a^\neq$. Since for Π^{RN} we have that for each $\rho \in P$, $I[x^o_\rho] = \bot$, the first temporal transition always involves $a^\neq$. It follows that $cost^{\text{RN}}(\pi_t) = cost(\pi^{\text{RN}}_{\text{EXP}})$. □

The handling of SW(τ), can be done by building on the translation EXP$^{\text{RN}}$. For the sake of brevity, we sketch the procedure. We introduce a novel numeric variable θ, namely the *context clock*, to track the duration for which a context remains active before switching. θ is increased by δ every time a temporal transition occurs without causing a context switch, i.e., when $a^=$ is applied. Moreover, the action $a^\neq$, associated with a context switch, is split into two mutually exclusive actions, i.e., $\{a^\neq_<, a^\neq_\geq\}$. $a^\neq_<$ is executed when the last context persisted for a quantity less than τ, i.e., $\langle \theta < \tau \rangle$. $a^\neq_\geq$ is executed in the opposite case. Both actions reset the counter to δ. All actions have a cost of 0, except for $a^\neq_<$, which incurs a unitary cost.

Combining Plan Cost Functions

Combining the introduced plan cost functions can be advantageous in some cases. For instance, minimising only the makespan may lead to energy-consuming plans, while minimising roughness alone could result in excessive response times. Thus, we consider a general case where $\mathfrak{C}$ is a linear combination of the discussed plan cost functions.

Let $\mathfrak{C}$ be defined as a linear combination of ψ, MS, and RN. We can encode $\Pi^{\mathfrak{C}}$ as a PDDL2.1 planning task, since all costs are additive. Specifically, we set $\mathfrak{C} = \alpha_0 \cdot \text{MS} + \alpha_1 \cdot \psi + \alpha_2 \cdot \text{RN}$. To generate the cost-preserving variant of EXP, we create the PDDL2.1 planning task $\Gamma^{\mathfrak{C}}_{\text{EXP}} = \langle \Pi^{\text{RN}}_{\text{EXP}}, c^{\mathfrak{C}}_{\text{EXP}} \rangle$, where $\Pi^{\text{RN}}_{\text{EXP}}$ is obtained by using EXP$^{\text{RN}}$ (Definition 11). The state-dependent cost function $c^{\mathfrak{C}}_{\text{EXP}}$ for an action $a^\star$ and a state s is defined as generalisation of the one from Definition 10:

$$c^{\mathfrak{C}}_{\text{EXP}}(a^\star, s) = \begin{cases} \alpha_1 \cdot c^{\psi}_{\text{EXP}}(a, s) & \text{if } a^\star = a' \in A'' \\ \alpha_0 \cdot \delta + \alpha_1 \cdot c^{\psi}(\mathcal{C}, s) & \text{if } a^\star = a^{\mathcal{C}}_{\text{RN-SIM}} \in A_{\text{RN-SIM}} \\ \alpha_2 & \text{if } a^\star = a^\neq \\ 0 & \text{otherwise} \end{cases}$$

where, in the first case, $a \in A_\Pi \cup A_{\text{SIMEV}} \cup \{a_{satu}\}$ is the action from which $a' \in A''$ was obtained.

Conclusion

To promote the use of PDDL+ in real-world applications, we formalised a set of cost functions to guide plan generation and fill the gap in formal and practical support for cost-based planning. We considered a diverse set of cost functions, ranging from traditional metrics like makespan to novel ones inspired by hybrid automata, that reflect the expressiveness of PDDL+. To assess their practicability, we integrated them within a translation-based approach.

Our starting point was an existing translation, namely EXP, from discrete-time PDDL+ to PDDL2.1. We modified this translation to make it *cost-aware*, meaning that the cost of a plan in the compiled task corresponds exactly to the cost in the original task. This result suggests, in principle, that it is possible to leverage cost functions during plan generation, rather than only for post hoc evaluation. Moreover, it enables the use of existing PDDL2.1 planning engines not only for goal achievement, but also for cost-sensitive planning in PDDL+.

However, while this translation provides a bridge between PDDL+ with cost functions and PDDL2.1, it also has some limitations. The translation we considered is exponential in the input size, which makes it impractical for larger planning instances. For this reason, we plan to investigate whether, and to what extent, cost preservation can be achieved through more compact translations. As a more scalable alternative, we intend to explore the POLY translation, which yields a feasible encoding but produces polynomially longer plans, a known drawback in plan generation.

Future work will focus on turning these results into practice. First, we plan to implement our framework and experimentally evaluate its performance. Second, we aim to extend existing planning engines with native support for cost functions, enabling more efficient and cost-aware reasoning. Finally, we are interested in studying how the definitions of cost functions can be adapted to support continuous-time PDDL+ semantics, which introduce different challenges compared to the discrete-time setting.

Acknowledgments

Francesco Percassi and Mauro Vallati were supported by a UKRI Future Leaders Fellowship [grant number MR/Z00005X/1]. Enrico Scala was supported by Climate Change AI project (No. IG-2023-174) and partially by PNRR MUR project PE0000013-FAIR, cascade funding call, ResilientPlan.

References

Aineto, D.; Scala, E.; Onaindia, E.; and Serina, I. 2023. Falsification of Cyber-Physical Systems Using PDDL+ Planning. In *ICAPS*, 2–6. AAAI Press.

Alaboud, F. K.; and Coles, A. 2019. Personalized Medication and Activity Planning in PDDL+. In *ICAPS*, 492–500. AAAI Press.

Alon, L.; Weitman, H.; Shleyfman, A.; and Kaminka, G. A. 2024. Planning to be Healthy: Towards Personalized Medication Planning. In *ECAI*, volume 392, 4232–4239. IOS Press.

Bogomolov, S.; Magazzeni, D.; Minopoli, S.; and Wehrle, M. 2015. PDDL+ Planning with Hybrid Automata: Foundations of Translating Must Behavior. In *ICAPS*, 42–46. AAAI Press.

Bogomolov, S.; Magazzeni, D.; Podelski, A.; and Wehrle, M. 2014. Planning as Model Checking in Hybrid Domains. In *AAAI*, 2228–2234. AAAI Press.

Bonassi, L.; Gerevini, A. E.; and Scala, E. 2024. Dealing with Numeric and Metric Time Constraints in PDDL3 via Compilation to Numeric Planning. In *AAAI*, 20036–20043. AAAI Press.

Capitanelli, A.; Maratea, M.; Mastrogiovanni, F.; and Vallati, M. 2018. On the manipulation of articulated objects in human-robot cooperation scenarios. *Robotics Auton. Syst.*, 109: 139–155.

Cardellini, M.; Maratea, M.; Vallati, M.; Boleto, G.; and Oneto, L. 2021. In-Station Train Dispatching: A PDDL+ Planning Approach. In *Proc. of ICAPS*, 450–458.

Cashmore, M.; Magazzeni, D.; and Zehtabi, P. 2020. Planning for Hybrid Systems via Satisfiability Modulo Theories. *J. Artif. Intell. Res.*, 67: 235–283.

Cesta, A.; and Oddi, A. 1996. DDL.1: A Formal Description of a Constraint Representation Language for Physical Domains. *New directions in AI planning*, 341–352.

Chen, J.; Williams, B. C.; and Fan, C. 2021. Optimal mixed discrete-continuous planning for linear hybrid systems. In *HSCC*, 8:1–8:12. ACM.

Fox, M.; Howey, R.; and Long, D. 2005. Validating Plans in the Context of Processes and Exogenous Events. In *Proceedings of The Twentieth National Conference on Artificial Intelligence Conference, AAAI 2005*, 1151–1156. AAAI Press / The MIT Press.

Fox, M.; and Long, D. 2003. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *J. Artif. Intell. Res.*, 20: 61–124.

Fox, M.; and Long, D. 2006. Modelling Mixed Discrete-Continuous Domains for Planning. *J. Artif. Intell. Res.*, 27: 235–297.

Fox, M.; Long, D.; and Magazzeni, D. 2012. Plan-based policies for efficient multiple battery load management. *J. Artif. Intell. Res.*, 44: 335–382.

Gerevini, A.; Saetti, A.; and Serina, I. 2008. An approach to efficient planning with numerical fluents and multi-criteria plan quality. *Artif. Intell.*, 172(8-9): 899–944.

Henzinger, T. A. 1996. The Theory of Hybrid Automata. In *LICS*, 278–292. IEEE Computer Society.

Henzinger, T. A.; Kopke, P. W.; Puri, A.; and Varaiya, P. 1998. What's Decidable about Hybrid Automata? *J. Comput. Syst. Sci.*, 57(1): 94–124.

Hespanha, J.; and Morse, A. 1999. Stability of switched systems with average dwell-time. In *IEEE CDC*, volume 3, 2655–2660.

Hoffmann, J. 2003. The Metric-FF Planning System: Translating "Ignoring Delete Lists" to Numeric State Variables. *J. Artif. Intell. Res.*, 20: 291–341.

Kiam, J. J.; Scala, E.; Jávega, M. R.; and Schulte, A. 2020. An AI-Based Planning Framework for HAPS in a Time-Varying Environment. In *ICAPS*, 412–420. AAAI Press.

Kouaiti, A. E.; Percassi, F.; Saetti, A.; McCluskey, T. L.; and Vallati, M. 2024. PDDL+ Models for Deployable yet Effective Traffic Signal Optimisation. In *ICAPS*, 168–177. AAAI Press.

Kuroiwa, R.; Shleyfman, A.; Piacentini, C.; Castro, M. P.; and Beck, J. C. 2022. The LM-Cut Heuristic Family for Optimal Numeric Planning with Simple Conditions. *J. Artif. Intell. Res.*, 75: 1477–1548.

Leofante, F.; Giunchiglia, E.; Ábrahám, E.; and Tacchella, A. 2020. Optimal Planning Modulo Theories. In *IJCAI*, 4128–4134. ijcai.org.

Li, D.; Scala, E.; Haslum, P.; and Bogomolov, S. 2018. Effect-Abstraction Based Relaxation for Linear Numeric Planning. In *IJCAI*, 4787–4793. ijcai.org.

McDermott, D.; Ghallab, M.; Howe, A.; Knoblock, C.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. PDDL - The Planning Domain Definition Language. *Technical Report*.

Mitra, S.; and Liberzon, D. 2004. Stability of hybrid automata with average dwell time: an invariant approach. In *CDC*, 1394–1399. IEEE.

Morse, A. S. 1996. Supervisory Control of Families of Linear Set-Point Controllers — Part I. Exact matching. *IEEE Trans. Autom. Control.*, 41(10): 1413–1431.

Penna, G. D.; Magazzeni, D.; and Mercorio, F. 2012. A universal planning system for hybrid domains. *Appl. Intell.*, 36(4): 932–959.

Percassi, F.; Bhatnagar, S.; Guo, R.; McCabe, K.; McCluskey, T. L.; and Vallati, M. 2023. An Efficient Heuristic for AI-based Urban Traffic Control. In *MT-ITS*, 1–6. IEEE.

Percassi, F.; Scala, E.; and Vallati, M. 2023. A Practical Approach to Discretised PDDL+ Problems by Translation to Numeric Planning. *J. Artif. Intell. Res.*, 76: 115–162.

Piotrowski, W.; and Perez, A. 2024. Real-World Planning with PDDL+ and Beyond. *arXiv preprint arXiv:2402.11901*.

Piotrowski, W.; Sher, Y.; Grover, S.; Stern, R.; and Mohan, S. 2023. Heuristic Search for Physics-Based Problems: Angry Birds in PDDL+. In *ICAPS*, 518–526. AAAI Press.

Piotrowski, W. M.; Fox, M.; Long, D.; Magazzeni, D.; and Mercorio, F. 2016. Heuristic Planning for PDDL+ Domains. In *IJCAI*, 3213–3219. IJCAI/AAAI Press.

Sanner, S. 2010. Relational Dynamic Influence Diagram Language (RDDL): Language description. *Unpublished ms. Australian National University*, 32: 27.

Say, B. 2023. Robust Metric Hybrid Planning in Stochastic Nonlinear Domains Using Mathematical Optimization. In *ICAPS*, 375–383. AAAI Press.

Scala, E.; Haslum, P.; Thiébaux, S.; and Ramírez, M. 2016. Interval-Based Relaxation for General Numeric Planning. In *ECAI*, volume 285 of *Frontiers in Artificial Intelligence and Applications*, 655–663. IOS Press.

Shin, J.; and Davis, E. 2005. Processes and continuous change in a SAT-based planner. *Artif. Intell.*, 166(1-2): 194–253.